Accelerating Natural Gradient with Higher-Order Invariance

Yang Song [1] **Jiaming Song** [1] **Stefano Ermon** [1]

Abstract

An appealing property of the natural gradient is that it is invariant to arbitrary differentiable reparameterizations of the model. However, this invariance property requires infinitesimal steps and is lost in practical implementations with small but finite step sizes. In this paper, we study invariance properties from a combined perspective of Riemannian geometry and numerical differential equation solving. We define the order of invariance of a numerical method to be its convergence order to an invariant solution. We propose to use higher-order integrators and geodesic corrections to obtain more invariant optimization trajectories. We prove the numerical convergence properties of geodesic corrected updates and show that they can be as computational efficient as plain natural gradient. Experimentally, we demonstrate that invariance leads to faster optimization and our techniques improve on traditional natural gradient in deep neural network training and natural policy gradient for reinforcement learning.

1. Introduction

Non-convex optimization is a key component of the success of deep learning. Current state-of-the-art training methods are usually variants of stochastic gradient descent (SGD), such as AdaGrad (Duchi et al., 2011), RMSProp (Hinton et al., 2012) and Adam (Kingma & Ba, 2015). While generally effective, performance of those first-order optimizers is highly dependent on the curvature of the optimization objective. When the Hessian matrix of the objective at the optimum has a large condition number, the problem is said to have pathological curvature (Martens, 2010; Sutskever et al., 2013), and first-order methods will have trouble in making progress. The curvature, however, depends on how the model is parameterized. There may be some equivalent way of parameterizing the same model which has better-behaved curvature and is thus easier to optimize with first-order methods. Model reparameterizations, such as good network architectures (Simonyan & Zisserman, 2014; He et al., 2016) and normalization techniques (LeCun et al., 2012; Ioffe & Szegedy, 2015; Salimans & Kingma, 2016) are often critical for the success of first-order methods.

The natural gradient (Amari, 1998) method takes a different perspective to the same problem. Rather than devising a different parameterization for first-order optimizers, it tries to make the optimizer itself invariant to reparameterizations by directly operating on the manifold of probabilistic models. This invariance, however, only holds in the idealized case of infinitesimal steps, *i.e.*, for continuous-time natural gradient descent trajectories on the manifold (Ollivier, 2013; 2015). Practical implementations with small but finite step size (learning rate) are only approximately invariant. Inspired by Newton-Raphson method, the learning rate of natural gradient method is usually set to values near 1 in real applications (Martens, 2010; 2014), leading to potential loss of invariance.

In this paper, we investigate invariance properties within the framework of Riemannian geometry and numerical differential equation solving. We observe that both the exact solution of the natural gradient dynamics and its approximation obtained with Riemannian Euler method (Bielecki, 2002) are invariant. We propose to measure the invariance of a numerical scheme by *studying its rate of convergence to those idealized truly invariant solutions*. It can be shown that the traditional natural gradient update (based on the forward Euler method) converges in first order. For improvement, we first propose to use a second-order Runge-Kutta integrator. Additionally, we introduce corrections based on the geodesic equation. We argue that the Runge-Kutta integrator converges to the exact solution in second order, and the method with geodesic corrections converges to the Riemannian Euler method in second order. Therefore, all the new methods have higher order of invariance, and experiments verify their faster convergence in deep neural

[1] Computer Science Department, Stanford University. Correspondence to: Yang Song <yangsong@cs.stanford.edu>, Jiaming Song <tsong@cs.stanford.edu>, Stefano Ermon <ermon@cs.stanford.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

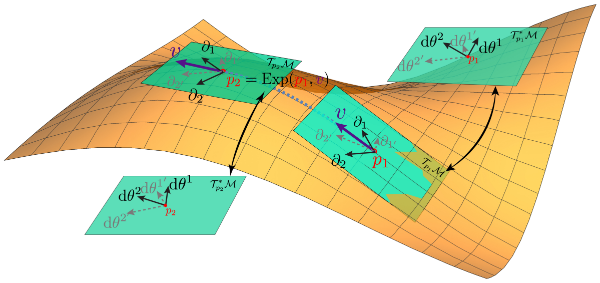

Figure 1. An illustration of Riemannian geometry concepts: tangent spaces, cotangent spaces, coordinate basis, dual coordinate basis, geodesics and the exponential map.

network training and policy optimization for deep reinforcement learning. Moreover, the geodesic correction update has a faster variant which keeps the second-order invariance while being roughly as time efficient as the original natural gradient update. Our new methods can be used as drop-in replacements in any situation where natural gradient may be used.

2. Preliminaries

2.1. Riemannian Geometry and Invariance

We use *Einstein's summation convention* throughout this paper to simplify formulas. The convention states that when any index variable appears twice in a term, once as a superscript and once as a subscript, it indicates summation of the term over all possible values of the index variable. For example, $a^\mu b_\mu \triangleq \sum_{\mu=1}^n a^\mu b_\mu$ when index variable $\mu \in [n]$.

Riemannian geometry is used to study intrinsic properties of differentiable manifolds equipped with metrics. The goal of this necessarily brief section is to introduce some key concepts related to the understanding of invariance. For more details, please refer to (Petersen, 2006) and (Amari et al., 1987).

In this paper, we describe a family of probabilistic models as a manifold. Roughly speaking, a *manifold* $\mathcal{M}$ of dimension n is a smooth space whose local regions resemble $\mathbb{R}^n$ (Carroll, 2004). Assume there exists a smooth mapping $\phi : \mathcal{M} \to \mathbb{R}^n$ in some neighborhood of p and for any $p \in \mathcal{M}$, $\phi(p)$ is the coordinate of p. As an example, if p is a parameterized distribution, $\phi(p)$ will refer to its parameters. There is a linear space associated with each $p \in \mathcal{M}$ called the *tangent space* $\mathcal{T}_p\mathcal{M}$. Each element $v \in \mathcal{T}_p\mathcal{M}$ is called a *vector*. For any tangent space $T_p\mathcal{M}$, there exists a dual space $T_p^*\mathcal{M}$ called the *cotagent space*, which consists of all linear real-valued functions on the tangent space. Each element v^* in the dual space $T_p^*\mathcal{M}$ is called a *covector*. Let $\phi(p) = (\theta^1, \theta^2, \cdots, \theta^n)$ be the coordinates of p, it can be shown that the set of operators $\{\frac{\partial}{\partial\theta^1}, \cdots, \frac{\partial}{\partial\theta^n}\}$ forms a basis for $\mathcal{T}_p\mathcal{M}$ and is called the *coordinate basis*. Similarly, the dual space admits the *dual coordinate basis* $\{\mathrm{d}\theta^1, \cdots, \mathrm{d}\theta^n\}$. These two sets of bases satisfy $\mathrm{d}\theta^\mu(\partial_\nu) = \delta^\mu_\nu$ where $\delta^\mu_\nu \triangleq \begin{cases} 1, \mu = \nu \\ 0, \mu \neq \nu \end{cases}$ is the Kronecker delta. Note that in this paper we abbreviate $\frac{\partial}{\partial\theta^\mu}$ to ∂_μ and often refer to an entity (*e.g.*, vector, covector and point on the manifold) with its coordinates.

Vectors and covectors are geometric objects associated with a manifold, which exist independently of the coordinate system. However, we rely on their **representations** w.r.t. some coordinate system for quantitative studies. Given a coordinate system, a vector $\mathbf{a}$ (covector $\mathbf{a}^*$) can be represented by its coefficients w.r.t. the coordinate (dual coordinate) bases, which we denote as a^μ (a_μ). Therefore, these coefficients depend on a specific coordinate system, and will change for different parameterizations. In order for those coefficients to represent coordinate-independent entities like vectors and covectors, their change should obey some appropriate transformation rules. Let the new coordinate system under a different parameterization be $\phi'(p) = (\xi^1, \cdots, \xi^n)$ and let the old one be $\phi(p) = (\theta^1, \cdots, \theta^n)$. It can be shown that the new coefficients of $\mathbf{a} \in \mathcal{T}_p\mathcal{M}$ will be given by $a^{\mu'} = a^\mu \frac{\partial\xi^{\mu'}}{\partial\theta^\mu}$, while the new coefficients of $\mathbf{a}^* \in \mathcal{T}^*_p\mathcal{M}$ will be determined by $a_{\mu'} = a_\mu \frac{\partial\theta^\mu}{\partial\xi^{\mu'}}$. Due to the difference of transformation rules, we say a^μ is *contravariant* while a_μ is *covariant*, as indicated by superscripts and subscripts respectively. In this paper, we only use Greek letters to denote contravariant / covariant components.

Riemannian manifolds are equipped with a positive definite metric tensor $g_p \in \mathcal{T}^*_p\mathcal{M} \otimes \mathcal{T}^*_p\mathcal{M}$, so that distances and angles can be characterized. The inner product of two vectors $\mathbf{a} = a^\mu\partial_\mu \in \mathcal{T}_p\mathcal{M}$, $\mathbf{b} = b^\nu\partial_\nu \in \mathcal{T}_p\mathcal{M}$ is defined as $\langle \mathbf{a}, \mathbf{b} \rangle \triangleq g_p(\mathbf{a}, \mathbf{b}) = g_{\mu\nu}\mathrm{d}\theta^\mu \otimes \mathrm{d}\theta^\nu(a^\mu\partial_\mu, b^\nu\partial_\nu) = g_{\mu\nu}a^\mu b^\nu$. For convenience, we denote the inverse of the metric tensor as $g^{\alpha\beta}$ using superscripts, *i.e.*, $g^{\alpha\beta}g_{\beta\mu} = \delta^\alpha_\mu$. The introduction of inner product induces a natural map from a tangent space to its dual space. Let $\mathbf{a} = a^\mu\partial_\mu \in \mathcal{T}_p\mathcal{M}$, its natural correspondence in $\mathcal{T}^*_p\mathcal{M}$ is the covector $\mathbf{a}^* \triangleq \langle \mathbf{a}, \cdot \rangle = a_\nu d\theta^\nu$. It can be shown that $a_\nu = a^\mu g_{\mu\nu}$ and $a^\mu = g^{\mu\nu}a_\nu$. We say the metric tensor relates the coefficients of a vector and its covector by lowering and raising indices, which effectively changes the transformation rule.

The metric structure makes it possible to define *geodesics* on the manifold, which are constant speed curves $\gamma : \mathbb{R} \to \mathcal{M}$ that are locally distance minimizing. Since the distances on manifolds are independent of parameterization, geodesics are invariant objects. Using a specific coordinate system, $\gamma(t)$ can be determined by solving the *geodesic equation*

$$\ddot{\gamma}^\mu + \Gamma^\mu_{\alpha\beta}\dot{\gamma}^\alpha\dot{\gamma}^\beta = 0, \tag{1}$$

where $\Gamma^{\mu}_{\alpha\beta}$ is the *Levi-Civita connection* defined by

$$\Gamma^{\mu}_{\alpha\beta} \triangleq \frac{1}{2} g^{\mu\nu}(\partial_\alpha g_{\nu\beta} + \partial_\beta g_{\nu\alpha} - \partial_\nu g_{\alpha\beta}). \tag{2}$$

Note that we use $\dot{\gamma}$ to denote $\frac{\mathrm{d}\gamma}{\mathrm{d}t}$ and $\ddot{\gamma}$ for $\frac{\mathrm{d}^2\gamma}{\mathrm{d}t^2}$.

Given $p \in \mathcal{M}$ and $v \in \mathcal{T}_p\mathcal{M}$, there exists a unique geodesic satisfying $\gamma(0) = p, \dot{\gamma}(0) = v$. If we follow the curve $\gamma(t)$ from $p = \gamma(0)$ for a unit time $\Delta t = 1$, we can reach another point $p' = \gamma(1)$ on the manifold. In this way, traveling along geodesics defines a map from $\mathcal{M} \times \mathcal{T}\mathcal{M}$ to $\mathcal{M}$ called *exponential map*

$$\operatorname{Exp}(p, v) \triangleq \gamma(1), \tag{3}$$

where $\gamma(0) = p$ and $\dot{\gamma}(0) = v$. By simple re-scaling we also have $\operatorname{Exp}(p, hv) = \gamma(h)$.

As a summary, we provide a graphical illustration of relevant concepts in Riemannian geometry in Figure 1. Here we emphasize again the important ontological difference between an object and its coordinate. The manifold itself, along with geodesics and vectors (covectors) in its tangent (cotangent) spaces is intrinsic and independent of coordinates. The coordinates need to be transformed correctly to describe the same objects and properties on the manifold under a different coordinate system. This is where invariance emerges.

2.2. Numerical Differential Equation Solvers

Let the ordinary differential equation (ODE) be $\dot{x}(t) = f(t, x(t))$, where $x(0) = a$ and $t \in [0, T]$. Numerical integrators try to trace $x(t)$ with iterative local approximations $\{x_k \mid k \in \mathbb{N}\}$.

We discuss several useful numerical methods in this paper. The forward Euler method updates its approximation by $x_{k+1} = x_k + hf(t_k, x_k)$ and $t_{k+1} = t_k + h$. It can be shown that as $h \to 0$, the error $\|x_k - x(t_k)\|$ can be bounded by $\mathcal{O}(h)$. The midpoint integrator is a Runge-Kutta method with $\mathcal{O}(h^2)$ error. Its update formula is given by $x_{k+1} = x_k + hf\left(t_k + \frac{1}{2}h, x_k + \frac{h}{2} f(t_k, x_k)\right)$, $t_{k+1} = t_k + h$. The Riemannian Euler method (see pp.3-6 in (Bielecki, 2002)) is a less common variant of the Euler method, which uses the Exponential map for its updates as $x_{k+1} = \operatorname{Exp}(x_k, hf(t_k, x_k))$, $t_{k+1} = t_k + h$. While having the same asymptotic error $\mathcal{O}(h)$ as forward Euler, it has more desirable invariance properties.

2.3. Revisiting Natural Gradient Method

Let $r_\theta(\mathbf{x}, \mathbf{t}) = p_\theta(\mathbf{t} \mid \mathbf{x})q(\mathbf{x})$ denote a probabilistic model parameterized by $\theta \in \Theta$, where $\mathbf{x}, \mathbf{t}$ are random variables, $q(\mathbf{x})$ is the marginal distribution of $\mathbf{x}$ and assumed to be fixed. Conventionally, $\mathbf{x}$ is used to denote the input and $\mathbf{t}$ represents its label. In a differential geometric framework, the set of all possible probabilistic models r_θ constitutes a manifold $\mathcal{M}$, and the parameter vector θ provides a coordinate system. Furthermore, the infinitesimal distance of probabilistic models can be measured by the Fisher information metric $g_{\mu\nu} = \mathbb{E}_{\mathbf{x}\sim q}\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}[\partial_\mu \log p_\theta(\mathbf{t} \mid \mathbf{x})\partial_\nu \log p_\theta(\mathbf{t} \mid \mathbf{x})]$. Let the loss function $L(r_\theta) = -\mathbb{E}_{\mathbf{x}\sim q}[\log p_\theta(\mathbf{l} \mid \mathbf{x})]$ be the expected negative log-likelihood, where $\mathbf{l}$ denotes the ground truth labels in the training dataset. Our learning goal is to find a model r_{θ^*} that minimizes the (empirical) loss $L(r_\theta)$.

The well known update rule of gradient descent $\theta^\mu_{k+1} = \theta^\mu_k - h\lambda\partial_\mu L(r_{\theta_k})$ can be viewed as approximately solving the (continuous time) ODE

$$\dot{\theta}^\mu = -\lambda\partial_\mu L(r_\theta) \tag{4}$$

with forward Euler method. Here λ is a time scale constant, h is the step size, and their product $h\lambda$ is the learning rate. Note that λ will only affect the "speed" but not trajectory of the system. It is notorious that the gradient descent ODE is not invariant to reparameterizations (Ollivier, 2013; Martens, 2010). For example, if we rescale θ^μ to $2\theta^\mu$, $\partial_\mu L(r_\theta)$ will be downscaled to $\frac{1}{2}\partial_\mu L(r_\theta)$. This is more evident from a differential geometric point of view. As can be verified by chain rule, $\dot{\theta}^\mu$ transforms contravariantly and can therefore be treated as a vector in $\mathcal{T}_p\mathcal{M}$, while $\partial_\mu L(r_\theta)$ transforms covariantly, thus being a covector in $\mathcal{T}^*_p\mathcal{M}$. Because Eq. (4) tries to relate objects in different spaces with different transformation rules, it is not an invariant relation.

Natural gradient alleviates this issue by approximately solving an invariant ODE. Recall that we can raise or lower an index given a metric tensor $g_{\mu\nu}$. By raising the index of $\partial_\mu L(r_\theta)$, the r.h.s. of the gradient descent ODE (Eq. (4)) becomes a vector in $\mathcal{T}_p\mathcal{M}$, which solves the type mismatch problem of Eq. (4). The new ODE

$$\dot{\theta}^\mu = -\lambda g^{\mu\nu}\partial_\nu L(r_\theta) \tag{5}$$

is now invariant, and the forward Euler approximation becomes $\theta^\mu_{k+1} = \theta^\mu_k - h\lambda g^{\mu\nu}\partial_\nu L(r_{\theta_k})$, which is the traditional natural gradient update (Amari, 1998).

3. Higher-order Integrators

If we could integrate the learning trajectory equation $\dot{\theta}^\mu = -\lambda g^{\mu\nu}\partial_\nu L$ exactly, the optimization procedure would be invariant to reparameterizations. However, the naïve linear update of natural gradient $\theta^\mu_{k+1} = \theta^\mu_k - h\lambda g^{\mu\nu}\partial_\nu L$ is only a forward Euler approximation, and can only converge to the invariant exact solution in first order. Therefore, a natural improvement is to use higher-order integrators to obtain a more accurate approximation to the exact solution.

As mentioned before, the midpoint integrator has second-order convergence and should be generally more accurate.

In our case, it becomes

$$\theta^\mu_{k+\frac{1}{2}} = \theta^\mu_k - \frac{1}{2} h\lambda g^{\mu\nu}(\theta_k)\partial_\nu L(r_{\theta_k}),$$
$$\theta^\mu_{k+1} = \theta^\mu_k - h\lambda g^{\mu\nu}(\theta_{k+\frac{1}{2}})\partial_\nu L(r_{\theta_{k+\frac{1}{2}}}).$$

where $g^{\mu\nu}(\theta_k), g^{\mu\nu}(\theta_{k+\frac{1}{2}})$ are the inverse metrics evaluated at θ_k and $\theta_{k+\frac{1}{2}}$ respectively. Since our midpoint integrator converges to the invariant natural gradient ODE solution in second order, it preserves higher-order invariance compared to the first-order Euler integrator used in vanilla natural gradient.

4. Riemannian Euler Method

For solving the natural gradient ODE (Eq. (5)), the Riemannian Euler method's update rule becomes

$$\theta^\mu_{k+1} = \operatorname{Exp}(\theta^\mu_k, -h\lambda g^{\mu\nu}\partial_\nu L(r_{\theta_k})), \tag{6}$$

where $\operatorname{Exp} : \{(p,v) \mid p \in \mathcal{M}, v \in \mathcal{T}_p\mathcal{M}\} \to \mathcal{M}$ is the exponential map as defined in Section 2.1. The solution obtained by Riemannian Euler method is invariant to reparameterizations, because Exp is a function independent of parameterization and for each step, the two arguments of Exp are both invariant.

4.1. Geodesic Correction

For most models, it is not tractable to compute Exp, since it requires solving the geodesic equation (1) exactly. Nonetheless, there are two numerical methods to approximate geodesics, with different levels of accuracy.

According to Section 2.1, $\operatorname{Exp}(\theta^\mu_k, -h\lambda g^{\mu\nu}\partial_\nu L(r_{\theta_k})) = \gamma^\mu_k(h)$, where γ^μ_k satisfies the geodesic equation (1) and

$$\gamma^\mu_k(0) = \theta^\mu_k$$
$$\dot\gamma^\mu_k(0) = -\lambda g^{\mu\nu}\partial_\nu L(r_{\theta_k}).$$

The first method for approximately solving $\gamma^\mu_k(t)$ ignores the whole geodesic equation and only uses information of first derivatives, giving

$$\gamma^\mu_k(h) \approx \theta^\mu_k + h\dot\gamma^\mu_k(0) = \theta^\mu_k - h\lambda g^{\mu\nu}\partial_\nu L,$$

which corresponds to the naïve natural gradient update rule.

The more accurate method leverages information of second derivatives from the geodesic equation (1). The result is

$$\gamma^\mu_k(h) \approx \theta^\mu_k + h\dot\gamma^\mu_k(0) + \frac{1}{2}h^2\ddot\gamma^\mu_k(0)$$
$$= \theta^\mu_k - h\lambda g^{\mu\nu}\partial_\nu L - \frac{1}{2}h^2\Gamma^\mu_{\alpha\beta}\dot\gamma^\alpha_k(0)\dot\gamma^\beta_k(0).$$

The additional second-order term given by the geodesic equation (1) reduces the truncation error to third-order. This corresponds to our new natural gradient update rule with *geodesic correction*, *i.e.*,

$$\theta^\mu_{k+1} = \theta^\mu_k + h\dot\gamma^\mu_k(0) - \frac{1}{2}h^2\Gamma^\mu_{\alpha\beta}\dot\gamma^\alpha_k(0)\dot\gamma^\beta_k(0), \tag{7}$$

where $\dot\gamma^\mu_k(0) = -\lambda g^{\mu\nu}\partial_\nu L(r_{\theta_k})$.

4.2. Faster Geodesic Correction

To obtain the second order term in the geodesic corrected update, we first need to compute $\dot\gamma_k(0)$, which requires inverting the Fisher information matrix. Then we have to plug in $\dot\gamma_k(0)$ and compute $\Gamma^\mu_{\alpha\beta}\dot\gamma^\alpha_k(0)\dot\gamma^\beta_k(0)$, which involves inverting the same Fisher information matrix again (see (2)). Matrix inversion (more precisely, solving the corresponding linear system) is expensive and it would be beneficial to combine the natural gradient and geodesic correction terms together and do only one inversion.

To this end, we propose to estimate $\dot\gamma_k(0)$ in $\Gamma^\mu_{\alpha\beta}\dot\gamma_k(0)^\alpha\dot\gamma_k(0)^\beta$ with $\dot\gamma_k(0) \approx (\theta_k - \theta_{k-1})/h$. Using this approximation and substituting (2) into (7) gives the following faster geodesic correction update rule:

$$\delta\theta^\mu_k = \lambda g^{\mu\nu}\cdot\Big[-\partial_\nu L(r_{\theta_k}) - \tag{8}$$
$$\frac{1}{4}h\lambda(\partial_\alpha g_{\nu\beta} + \partial_\beta g_{\nu\alpha} - \partial_\nu g_{\alpha\beta})\delta\theta^\alpha_{k-1}\delta\theta^\beta_{k-1}\Big]$$
$$\theta^\mu_{k+1} = \theta^\mu_k + h\delta\theta^\mu_k, \tag{9}$$

which only involves one inversion of the Fisher information matrix.

4.3. Convergence Theorem

We summarize the convergence properties of geodesic correction and its faster variant in the following general theorem.

Theorem 1 (Informal). *Consider the initial value problem $\dot x = f(t, x(t)), x(0) = a, 0 \le t \le T$. Let the interval $[0, T]$ be subdivided into n equal parts by the grid points $0 = t_0 < t_1 < \cdots < t_n = T$, with the grid size $h = T/n$. Denote x_k and $\hat x_k$ as the numerical solution given by geodesic correction and its faster version respectively. Define the error e_k at each grid point x_k by $e_k = x'_k - x_k$, and $\hat e_k = x'_k - \hat x_k$, where x'_k is the numerical solution given by Riemannian Euler method. Then it follows that*

$$\|e_k\| \le \mathcal{O}(h^2) \quad \textit{and} \quad \|\hat e_k\| \le \mathcal{O}(h^2), h \to 0, \forall k \in [n].$$

As a corollary, both Euler's update with geodesic correction and its faster variant converge to the solution of ODE in 1st order.

Proof. Please refer to Appendix A for a rigorous statement and detailed proof. □

The statement of Theorem 1 is general enough to hold beyond the natural gradient ODE (Eq. (5)). It shows that both geodesic correction and its faster variant converge to the invariant Riemannian Euler method in 2nd order. In contrast, vanilla forward Euler method, as used in traditional natural gradient, is a first order approximation of Riemannian Euler method. In this sense, geodesic corrected updates preserve higher-order invariance.

5. Geodesic Correction for Neural Networks

Adding geodesic correction requires computing the Levi-Civita connection $\Gamma^{\alpha}_{\mu\nu}$ (see (2)), which usually involves second-order derivatives. This is to the contrast of natural gradient, where the computation of Fisher information matrix only involves first-order derivatives of outputs. In this section, we address the computational issues of geodesic correction in optimizing deep neural networks.

In order to use natural gradient for neural network training, we first need to convert neural networks to probabilistic models. A feed-forward network can be treated as a conditional distribution $p_\theta(\mathbf{t} \mid \mathbf{x})$. For regression networks, $p_\theta(\mathbf{t} \mid \mathbf{x})$ is usually a family of multivariate Gaussians. For classification networks, $p_\theta(\mathbf{t} \mid \mathbf{x})$ usually becomes a family of categorical distributions. The joint probability density is $q(\mathbf{x})p_\theta(\mathbf{t} \mid \mathbf{x})$, where $q(\mathbf{x})$ is the data distribution and is usually approximated with the empirical distribution.

The first result in this section is the analytical formula of the Levi-Civita connection of neural networks.

Proposition 1. *The Levi-Civita connection of a neural network model manifold is given by*

$$\begin{aligned}\Gamma^{\mu}_{\alpha\beta} = g^{\mu\nu}&\mathbb{E}_{q(\mathbf{x})}\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\\ &\Big\{\partial_\nu \log p_\theta(\mathbf{t} \mid \mathbf{x})\Big[\partial_\alpha\partial_\beta \log p_\theta(\mathbf{t} \mid \mathbf{x}) +\\ &\qquad \frac{1}{2}\partial_\alpha \log p_\theta(\mathbf{t} \mid \mathbf{x})\partial_\beta \log p_\theta(\mathbf{t} \mid \mathbf{x})\Big]\Big\} \qquad (10)\end{aligned}$$

Proof. In Appendix A. □

We denote the outputs of a neural network as $\mathbf{y}(\mathbf{x}, \theta) = (y_1, y_2, \cdots, y_o)$, which is an o-dimensional vector if there are o output units. In this paper, we assume that $\mathbf{y}(\mathbf{x}, \theta)$ are the values *after* final layer activation (*e.g.*, softmax). For typical loss functions, the expectation with respect to the corresponding distributions can be calculated analytically. Specifically, we instantiate the Levi-Civita connection for model distributions induced by three common losses and summarize them in the following proposition.

Proposition 2. *For the squared loss, we have*

$$\begin{aligned}p_\theta(\mathbf{t} \mid \mathbf{x}) &= \prod_{i=1}^{o} \mathcal{N}(t_i \mid y_i, \sigma^2)\\ g_{\mu\nu} &= \frac{1}{\sigma^2}\sum_{i=1}^{o} \mathbb{E}_{q(\mathbf{x})}[\partial_\mu y_i \partial_\nu y_i]\\ \Gamma^{\mu}_{\alpha\beta} &= \frac{1}{\sigma^2}\sum_{i=1}^{o} g^{\mu\nu}\mathbb{E}_{q(\mathbf{x})}[\partial_\nu y_i \partial_\alpha\partial_\beta y_i]\end{aligned}$$

For the binary cross-entropy loss, we have

$$\begin{aligned}p_\theta(\mathbf{t} \mid \mathbf{x}) &= \prod_{i=1}^{o} y_i^{t_i}(1-y_i)^{1-t_i}\\ g_{\mu\nu} &= \sum_{i=1}^{o} \mathbb{E}_{q(\mathbf{x})}\left[\frac{1}{y_i(1-y_i)} \cdot \partial_\mu y_i \partial_\nu y_i\right]\\ \Gamma^{\mu}_{\alpha\beta} &= g^{\mu\nu}\sum_{i=1}^{o} \mathbb{E}_{q(\mathbf{x})}\Bigg[\frac{2y_i - 1}{2y_i^2(1-y_i)^2} \cdot \partial_\nu y_i \partial_\alpha y_i \partial_\beta y_i\\ &\qquad + \frac{1}{y_i(1-y_i)} \cdot \partial_\nu y_i \partial_\alpha\partial_\beta y_i\Bigg].\end{aligned}$$

In the case of multi-class cross-entropy loss, we have

$$\begin{aligned}p_\theta(\mathbf{t} \mid \mathbf{x}) &= \prod_{i=1}^{o} y_i^{t_i}\\ g_{\mu\nu} &= \frac{1}{\sigma^2}\sum_{i=1}^{o} \mathbb{E}_{q(\mathbf{x})}\left[\frac{1}{y_i} \cdot \partial_\mu y_i \partial_\nu y_i\right]\\ \Gamma^{\mu}_{\alpha\beta} &= g^{\mu\nu}\sum_{i=1}^{o} \mathbb{E}_{q(\mathbf{x})}\Bigg[\frac{1}{y_i} \cdot \partial_\nu y_i \partial_\alpha\partial_\beta y_i\\ &\qquad - \frac{1}{2y_i^2} \cdot \partial_\nu y_i \partial_\alpha y_i \partial_\beta y_i\Bigg].\end{aligned}$$

Proof. In Appendix B. □

For geodesic correction, we only need to compute connection-vector products $\Gamma^{\mu}_{\alpha\beta}\dot{\gamma}^{\alpha}\dot{\gamma}^{\beta}$. This can be done with a similar idea to Hessian-vector products (Pearlmutter, 1994), for which we provide detailed derivations and pseudocodes in Appendix C. It can also be easily handled with automatic differentiation frameworks. We discuss some practical considerations on how to apply them in real cases in Appendix D.

6. Related Work

The idea of using the geodesic equation to accelerate gradient descent on manifolds was first introduced in Transtrum et al. (2011). However, our geodesic correction has several important differences. Our framework is generally applicable to all probabilistic models. This is to be contrasted

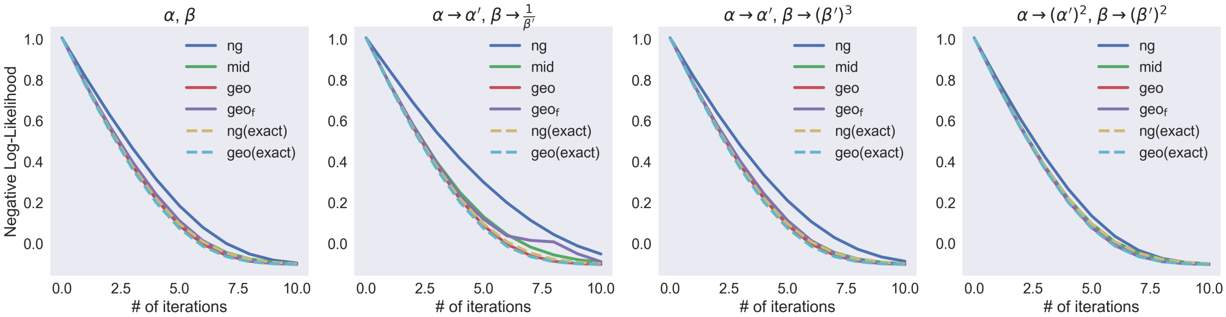

Figure 2. The effect of re-parameterizations on algorithms fitting a univariate Gamma distribution. Titles indicate which parameterization was used.

with "geodesic acceleration" in Transtrum et al. (2011) and Transtrum & Sethna (2012), which can only be applied to nonlinear least squares. Additionally, our geodesic correction is motivated from the perspective of preserving higher-order invariance, while in Transtrum & Sethna (2012) it is motivated as a higher-order correction to the Gaussian-Newton approximation of the Hessian under the so-called "small-curvature assumption". We discuss and evaluate empirically in Appendix F why the small-curvature approximation does not hold for training deep neural networks.

There has been a resurgence of interest in applying natural gradient to neural network training. Martens (2010) and Martens & Sutskever (2011) show that Hessian-Free optimization, which is equivalent to natural gradient method in important cases in practice (Pascanu & Bengio, 2013; Martens, 2014), is able to obtain state-of-the-art results in optimizing deep autoencoders and RNNs. To scale up natural gradient, some approximations for inverting the Fisher information matrix have been recently proposed, such as Krylov subspace descent (Vinyals & Povey, 2012), FANG (Grosse & Salakhutdinov, 2015) and K-FAC (Martens & Grosse, 2015; Grosse & Martens, 2016; Ba et al., 2017).

7. Experimental Evaluations

In this section, we demonstrate the benefit of respecting higher-order invariance through experiments on synthetic optimization problems, deep neural net optimization tasks and policy optimization in deep reinforcement learning.

Algorithms have abbreviated names in figures. We use "**ng**" to denote the basic natural gradient, "**geo**" to denote the one with geodesic correction, "**geo$_\mathbf{f}$**" to denote the faster geodesic correction, and "**mid**" to abbreviate natural gradient update using midpoint integrator.

7.1. Invariance

In this experiment, we investigate the effect of invariance under different parameterizations of the same objective. We test different algorithms on fitting a univariate Gamma distribution via Maximum Log-Likelihood. The problem is simple—we can calculate the Fisher information metric and corresponding Levi-Civita connection accurately. Moreover, we can use ODE-solving software to numerically integrate the continuous natural gradient equation and calculate the exponential map used in Riemannian Euler method.

The pdf of Gamma distribution is

$$p(x \mid \alpha, \beta) = \Gamma(x; \alpha, \beta) \triangleq \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x},$$

where α, β are shape and rate parameters. Aside from the original parameterization, we test three others: 1) $\alpha = \alpha', \beta = 1/\beta'$; 2) $\alpha = \alpha', \beta = (\beta')^3$ and 3) $\alpha = (\alpha')^2, \beta = (\beta')^2$, where α', β' are new parameters. We generate 10000 synthetic data points from $\Gamma(X; 20, 20)$. During training, α and β are initialized to 1 and the learning rate is fixed to 0.5.

We summarize the results in Figure 2. Here "**ng(exact)**" is obtained by numerically integrating (5), and "**geo(exact)**" is obtained using Riemannian Euler method with a numerically calculated exponential map function. As predicted by the theory, both methods are exactly invariant under all parameterizations. From Figure 2 we observe that the vanilla natural gradient update is not invariant under re-parameterizations, due to its finite step size. We observe that our midpoint natural gradient method and geodesic corrected algorithms are more resilient to re-parameterizations, and all lead to accelerated convergence of natural gradient.

7.2. Training Deep Neural Nets

We test our algorithms on deep autoencoding and classification problems. The datasets are CURVES, MNIST and FACES, all of which contain small gray-scale images of various objects, *i.e.*, synthetic curves, hand-written digits and hu-

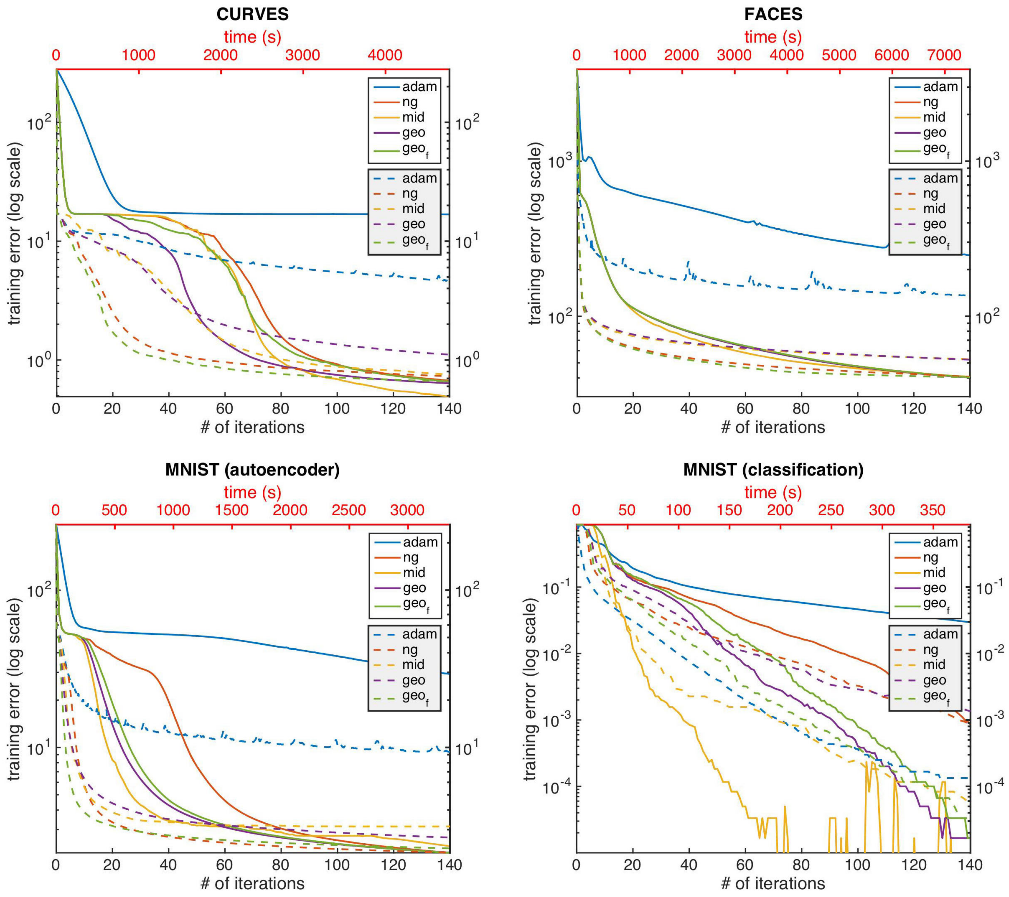

Figure 3. Training deep auto-encoders and classifiers with different acceleration algorithms. Solid lines show performance against number of iterations (bottom axes) while dashed lines depict performance against running time (top axes).

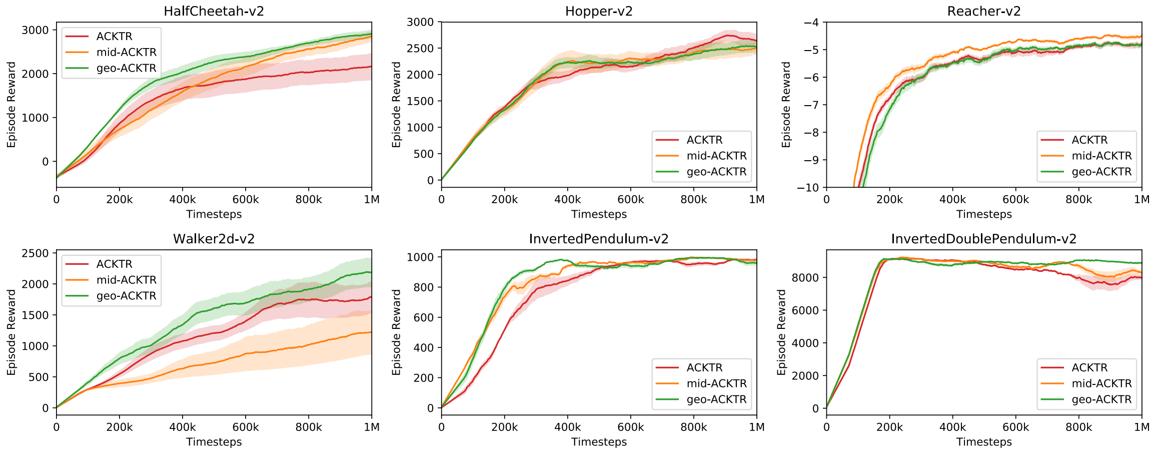

Figure 4. Sample efficiency of model-free reinforcement learning on continuous control tasks (Todorov et al., 2012). Titles indicate the environment used in OpenAI Gym (Brockman et al., 2016).

man faces. Since all deep networks use fully-connected layers and sigmoid activation functions, the tasks are non-trivial to solve even for modern deep learning optimizers, such as Adam (Kingma & Ba, 2015). Due to the high difficulty of this task, it has become a standard benchmark for neural network optimization algorithms (Hinton & Salakhutdinov, 2006; Martens, 2010; Vinyals & Povey, 2012; Sutskever et al., 2013; Martens & Grosse, 2015). Since these tasks only involve squared loss and binary cross-entropy, we additionally test multi-class cross-entropy on a MNIST classification task. Additional details can be found in Appendix E.1.

Figure 3 summarizes our results on all three datasets. Here "**ng**" is the natural gradient method (Hessian-Free) as implemented in Martens (2010). For completeness, we also add Adam (Kingma & Ba, 2015) into comparison and denote it as "**adam**". The training error reported for all datasets is the squared reconstruction error. Since the test error traces the training error very well, we only report the result of training error due to space limitation. It is clear that all acceleration methods lead to per iteration improvements compared to naïve natural gradient. It is also remarkable that the performance of "**geo**$_{\mathbf{f}}$", while being roughly half as expensive as "**geo**" per iteration, does not degrade too much compared to "**geo**". For performance comparisons with respect to time, "**geo**$_{\mathbf{f}}$" is usually the best (or comparable to the best). "**mid**" and "**geo**" are relatively slower, since they need roughly twice as much computation per iteration as "**ng**". Nonetheless, "**mid**" still has the best time performance for MNIST classification task.

We hereby emphasize again that geodesic correction methods are not aimed for providing more accurate solutions of the natural gradient ODE. Instead, they are higher-order approximations of an invariant solution (obtained by Riemannian Euler method), which itself is a first-order approximation to the exact solution. The improvements of both geodesic correction and midpoint integrator in Figure 2 and Figure 3 confirm our intuition that preserving higher-order invariance can accelerate natural gradient optimization.

7.3. Model-free Reinforcement Learning for Continuous Control

Finally, we evaluate our methods in reinforcement learning over six continuous control tasks (Todorov et al., 2012). Specifically, we consider improving the algorithm of ACKTR (Wu et al., 2017), an efficient variant of natural policy gradient (Kakade, 2002) which uses Kronecker factors (Martens & Grosse, 2015) to approximately compute the inverse of Fisher information matrix. For these methods, we evaluate sample efficiency (expected rewards per episode reached within certain numbers of interactions); in robotics tasks the cost of simulation often dominates the cost of the reinforcement learning algorithm, so requiring less interactions to achieve certain performance has higher priority than lower optimization time per iteration. Therefore we only test midpoint integrator and geodesic correction method for improving ACKTR, and omit the faster geodesic correction because of its less accurate approximation.

Figure 4 describes the results on the continuous control tasks, where we use "**mid-**" and "**geo-**" to denote our midpoint integrator and geodesic correction methods for ACKTR respectively. In each environment, we consider the same constant learning rate schedule for all three methods (detailed settings in Appendix E.2). While the Fisher information matrices are approximated via Kronecker factors, our midpoint integrator and geodesic correction methods are still able to outperform ACKTR in terms of sample efficiency in most of the environments. This suggests that preserving higher-order invariance could also benefit natural policy gradients in reinforcement learning, and our methods can be scaled to large problems via approximations of Fisher information matrices.

8. Conclusion

Our contributions in this paper can be summarized as:

- We propose to measure the invariance of numerical schemes by comparing their convergence to idealized invariant solutions.
- To the best of our knowledge, we are the first to use midpoint integrators for natural gradient optimization.
- Based on Riemannian Euler method, we introduce geodesic corrected updates. Moreover, the faster geodesic correction has comparable time complexity with vanilla natural gradient. Computationally, we also introduce new backpropagation type algorithms to compute connection-vector products. Theoretically, we provide convergence proofs for both types of geodesic corrected updates.
- Experiments confirm the benefits of invariance and demonstrate faster convergence and improved sample efficiency of our proposed algorithms in supervised learning and reinforcement learning applications.

For future research, it would be interesting to perform a thorough investigation over applications in reinforcement learning, and studying faster variants and more efficient implementations of the proposed acceleration algorithms.

Acknowledgements

The authors would like to thank Jonathan McKinney for helpful discussions. This work was supported by NSF grants #1651565, #1522054, #1733686, Toyota Research Institute, Future of Life Institute, and Intel.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Amari, S.-I. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Amari, S.-I., Barndorff-Nielsen, O., Kass, R., Lauritzen, S., Rao, C., et al. Differential geometrical theory of statistics. In *Differential geometry in statistical inference*, pp. 19–94. Institute of Mathematical Statistics, 1987.

Ba, J., Grosse, R., and Martens, J. Distributed second-order optimization using kronecker-factored approximations. 2017.

Bielecki, A. Estimation of the euler method error on a riemannian manifold. *Commun. Numer. Meth. Engng.*, 18(11):757–763, 1 November 2002.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Carroll, S. M. Spacetime and geometry. an introduction to general relativity. *Spacetime and geometry/Sean Carroll. San Francisco, CA, USA: Addison Wesley, ISBN 0-8053-8732-3, 2004, XIV+ 513 pp.*, 1, 2004.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pp. 2933–2941, 2014.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Openai baselines. `https://github.com/openai/baselines`, 2017.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Grosse, R. and Martens, J. A kronecker-factored approximate fisher matrix for convolution layers. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 573–582, 2016.

Grosse, R. and Salakhutdinov, R. Scaling up natural gradient by sparsely factorizing the inverse fisher matrix. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 2304–2313, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hinton, G., Srivastava, N., and Swersky, K. Lecture 6a overview of mini–batch gradient descent. 2012.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *science*, 313 (5786):504–507, 2006.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Kakade, S. M. A natural policy gradient. In *Advances in neural information processing systems*, pp. 1531–1538, 2002.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Marquardt, D. W. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

Martens, J. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 735–742, 2010.

Martens, J. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

Martens, J. and Grosse, R. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 2408–2417, 2015.

Martens, J. and Sutskever, I. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1033–1040, 2011.

Ollivier, Y. Riemannian metrics for neural networks i: feedforward networks. *arXiv preprint arXiv:1303.0818*, 2013.

Ollivier, Y. Riemannian metrics for neural networks ii: recurrent networks and learning symbolic data sequences. *Information and Inference*, 4(2):154–193, 2015.

Pascanu, R. and Bengio, Y. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Pearlmutter, B. A. Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160, 1994.

Petersen, P. *Riemannian geometry*, volume 171. Springer, 2006.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 901–909, 2016.

Schraudolph, N. N. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *Proceedings of The 30th International Conference on Machine Learning*, pp. 1139–1147, 2013.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Transtrum, M. K. and Sethna, J. P. Geodesic acceleration and the small-curvature approximation for nonlinear least squares. *arXiv preprint arXiv:1207.4999*, 2012.

Transtrum, M. K., Machta, B. B., and Sethna, J. P. Geometry of nonlinear least squares with applications to sloppy models and optimization. *Physical Review E*, 83 (3):036701, 2011.

Vinyals, O. and Povey, D. Krylov subspace descent for deep learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 1261–1268, 2012.

Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pp. 5285–5294, 2017.